Project Update: A New Foundation of Attack Trees in Linear Logic

Harley Eades III

Computer Science
Augusta University
harley.eades@gmail.com

Abstract. In this short paper I provide an update on the status of newly funded research project investigating founding attack trees in the resource conscious theory called linear logic.

1 Introduction

What is a mathematical model of attack trees? There have been numerous proposed answers to this question. Some examples are propositional logic, multisets, directed acyclic graphs, source sink graphs (or parallel-series pomsets), Petri nets, and Markov processes. Is there a unifying foundation in common to each of these proposed models? Furthermore, can this unifying foundation be used to further the field of attack trees and build new tools for conducting threat analysis?

The answer to the first question is positive, but the answer to the second question is open. Each of the proposed models listed above have something in common. They can all be modeled in some form of a symmetric monoidal category[1] [6,1,2,3] – for the definition of a symmetric monoidal category see Appendix A. That is all well and good, but what can we gain from monoidal categories?

Monoidal categories are a mathematical model of linear logic as observed through the beautiful Curry-Howard-Lambek correspondence [5]. In linear logic every hypothesis must be used exactly once, and hence, if we view a hypothesis as a resource, then this property can be stated as every resource must be consumed. This linearity property is achieved by removing the structural rules for weakening and contraction from classical or intuitionistic logic. Thus, from a resource perspective, resources cannot be spontaneously created or duplicated – hence, propositional logic can be viewed as a degenerate, from a resource perspective, form of linear logic.

Multisets and Petri nets both capture the idea that the nodes of an attack tree capture both the attack action and the state – the resource – of the system being analyzed. As it turns out, linear logic has been shown to be a logical foundation for multisets [6] and Petri Nets [1]. Thus, linear logic has the ability

[1] I provide a proof that the category of source sink graphs is monoidal in Appendix B.

to model the state as well as attack actions of the goals of an attack tree. We propose that linear logic be used as the logical foundation of attack trees.

This projects[2] main goal is to determine the suitability of linear logic as a foundation for attack trees. The type of attack trees we consider in this paper are attack trees with sequential composition similar to Jhawar et al. [4]. Attack trees consist of two layers: the logical layer or process tree, and the quantitative layer. One interesting aspect of our proposed foundation is that the logical layer can come to the aid of the quantitative layer.

At the logical layer the base attacks of an attack tree will correspond to atomic formulas in linear logic, and each branching node of an attack tree will correspond to a binary operator in linear logic. Adding costs corresponds to annotating the atomic formulas with some base cost, and then annotating the binary operators with a cost computed from the costs of their respective left operand (left subtree) and right operand (right subtree). The most interesting aspect of adding costs is that computing the costs at the branching nodes is done during the construction of a derivation using the inference rules of the logic.

The inference rules of linear logic provide a number of benefits when constructing attack trees. First, the inference rules will certify that an attack tree is constructed correctly and all costs on branching nodes will be computed from the costs of the left and right subtrees. On top of that the inference rules offer a means of proving when two attack trees are equivalent. Leveraging the fact that language of attack trees corresponds to a very simple fragment of linear logic, e.g. linear implication is not needed, I conjecture that proving equivalence of attack trees can be automated. Thus, this could be used in practice to certify the correctness of transformations of attack trees maintaining the resource based semantics.

2 Attack Trees

In this section I introduce attack trees with sequential composition first proposed by Jhawar et al. [4]. One of the projects ultimate goals is to extend attack trees with even more operators driven by our choice of semantics, but we leave this to future work. The syntax for attack trees is defined in the following definition.

Definition 1. *Suppose* B *is a set of base attack,* C *is a set of costs,* $\Sigma = \{\mathsf{op}_{\odot}, \mathsf{op}_{\rhd}, \mathsf{op}_{\sqcup}\}$ *is a set of binary operations on* C*, and* $I = \{i_{\odot}, i_{\rhd}, i_{\sqcup}\}$ *is a set of identities, such that, each* $l \in \{\mathsf{op}_{\odot}, \mathsf{op}_{\sqcup}\}$*,* $\mathsf{op}_l \in \Sigma$ *and* $i_l \in I$ *make* $(\mathsf{C}, \mathsf{op}_l, i_l)$ *a commutative monoid, and* $(\mathsf{C}, \mathsf{op}_{\rhd}, i_{\rhd})$ *a non-communicative monoid.*

The following defines the syntax of ***Attack Trees*** *given an assignment* $\mathsf{c} : \mathsf{B} \to \mathsf{C}$ *of base attacks to costs:*

$$t ::= (b, \mathsf{c}(b)) \mid t_1 \sqcup_{\mathsf{op}_{\sqcup}} t_2 \mid t_1 \odot_{\mathsf{op}_{\odot}} t_2 \mid t_1 \rhd_{\mathsf{op}_{\rhd}} t_2$$

[2] This material is based upon work supported by the National Science Foundation CRII CISE Research Initiation grant, “CRII:SHF: A New Foundation for Attack Trees Based on Monoidal Categories“, under Grant No. 1565557.

I denote unsynchronized parallel composition of attacks by $t_1 \odot_{\mathsf{op}_\odot} t_2$*, choice between attacks by* $t_1 \sqcup_{\mathsf{op}_\sqcup} t_2$*, and sequential composition of attacks by* $t_1 \rhd_{\mathsf{op}_\rhd} t_2$*.*

The following rules define the attack tree equivalence relation (omitting operation annotations for readability):

$$\frac{}{(t_1 \,\mathsf{op}\, t_2)\,\mathsf{op}\, t_3 = t_1 \,\mathsf{op}(t_2 \,\mathsf{op}\, t_3)}\ \textsc{assoc} \qquad \frac{}{(t_1 \sqcup t_2) \odot t_3 = (t_1 \odot t_3) \sqcup (t_2 \odot t_3)}\ \textsc{dist}_1$$

$$\frac{}{t_1 \,\mathsf{op_S}\, t_2 = t_2 \,\mathsf{op_S}\, t_1}\ \textsc{sym} \qquad \frac{}{(t_1 \sqcup t_2) \rhd t_3 = (t_1 \rhd t_3) \sqcup (t_2 \rhd t_3)}\ \textsc{dist}_2$$

where $\mathsf{op} \in \{\odot, \rhd, \sqcup\}$ *and* $\mathsf{op_S} \in \{\odot, \sqcup\}$*.*

The equivalence relation is essentially the equivalence given in Jhaware et al. [4] – Theorem 1. Conducting an analysis of the threat potential of a complex system might yield an attack tree that must be rearranged to be understood, but this rearrangement should not compromise the meaning of the tree.

We wish to treat attack trees as formulas of a linear logic where $t_1 \sqcup_{\mathsf{op}_\sqcup} t_2$ is the commutative tensor product and $t_1 \rhd_{\mathsf{op}_\rhd} t_2$ is the non-communicative tensor product. Incorporating these two operators within the same linear logic is an interesting question in itself.

3 A Linear Logic for Attack Trees

References

1. Carolyn Brown, Doug Gurr, and Valeria Paiva. A linear specification language for petri nets. *DAIMI Report Series*, 20(363), 1991.
2. Marcelo Fiore and Marco Devesas Campos. *Computation, Logic, Games, and Quantum Foundations. The Many Facets of Samson Abramsky: Essays Dedicated to Samson Abramsky on the Occasion of His 60th Birthday*, chapter The Algebra of Directed Acyclic Graphs, pages 37–51. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.
3. Luisa Francesco Albasini, Nicoletta Sabadini, and Robert F. C. Walters. The compositional construction of markov processes. *Applied Categorical Structures*, 19(1):425–437, 2010.
4. Ravi Jhawar, Barbara Kordy, Sjouke Mauw, SaÅ!'a RadomiroviÄ, and Rolando Trujillo-Rasua. Attack trees with sequential conjunction. In Hannes Federrath and Dieter Gollmann, editors, *ICT Systems Security and Privacy Protection*, volume 455 of *IFIP Advances in Information and Communication Technology*, pages 339–353. Springer International Publishing, 2015.
5. Paul-André Melliès. Categorical semantics of linear logic. In Pierre-Louis Curien, Hugo Herbelin, Jean-Louis Krivine, and Paul-André Melliès, editors, *Interactive models of computation and program behaviour*. Panoramas et Synthèses 27, Société Mathématique de France, 2009.
6. A Tzouvaras. The linear logic of multisets. *Logic Journal of IGPL*, 6(6):901–916, 1998.

Appendix

A Symmetric Monoidal Categories

This appendix provides the definitions of both categories in general, and, in particular, symmetric monoidal closed categories. We begin with the definition of a category:

Definition 2. *A* ***category****, $\mathcal{C}$, consists of the following data:*

- *A set of objects $\mathcal{C}_0$, each denoted by A, B, C, etc.*
- *A set of morphisms $\mathcal{C}_1$, each denoted by f, g, h, etc.*
- *Two functions* src*, the source of a morphism, and* tar*, the target of a morphism, from morphisms to objects. If $\mathsf{src}(f) = A$ and $\mathsf{tar}(f) = B$, then we write $f : A \to B$.*
- *Given two morphisms $f : A \to B$ and $g : B \to C$, then the morphism $f;g : A \to C$, called the composition of f and g, must exist.*
- *For every object $A \in \mathcal{C}_0$, the there must exist a morphism $\mathsf{id}_A : A \to A$ called the identity morphism on A.*
- *The following axioms must hold:*
  - *(Identities) For any $f : A \to B$, $f;\mathsf{id}_B = f = \mathsf{id}_A;f$.*
  - *(Associativity) For any $f : A \to B$, $g : B \to C$, and $h : C \to D$, $(f;g);h = f;(g;h)$.*

Categories are by definition very abstract, and it is due to this that makes them so applicable. The usual example of a category is the category whose objects are all sets, and whose morphisms are set-theoretic functions. Clearly, composition and identities exist, and satisfy the axioms of a category. A second example is preordered sets, $(A, \leq)$, where the objects are elements of A and a morphism $f : a \to b$ for elements $a, b \in A$ exists iff $a \leq b$. Reflexivity yields identities, and transitivity yields composition.

Symmetric monoidal categories pair categories with a commutative monoid like structure called the tensor product.

Definition 3. *A* ***symmetric monoidal category (SMC)*** *is a category, $\mathcal{M}$, with the following data:*

- *An object I of $\mathcal{M}$,*
- *A bi-functor $\otimes : \mathcal{M} \times \mathcal{M} \to \mathcal{M}$,*
- *The following natural isomorphisms:*

$$\begin{aligned}
&\lambda_A : I \otimes A \to A\\
&\rho_A : A \otimes I \to A\\
&\alpha_{A,B,C} : (A \otimes B) \otimes C \to A \otimes (B \otimes C)
\end{aligned}$$

- *A symmetry natural transformation:*

$$\beta_{A,B} : A \otimes B \to B \otimes A$$

– *Subject to the following coherence diagrams:*

$$\begin{tikzcd}
((A \otimes B) \otimes C) \otimes D \arrow[r, "\alpha_{A,B,C} \otimes \mathsf{id}_D"] \arrow[d, "\alpha_{A\otimes B,C,D} g"] & (A \otimes (B \otimes C)) \otimes D \arrow[dd, "\alpha_{A,B\otimes C,D}"] \\
(A \otimes B) \otimes (C \otimes D) \arrow[d, "\alpha_{A,B,C\otimes D}"] & \\
A \otimes (B \otimes (C \otimes D)) & A \otimes ((B \otimes C) \otimes D) \arrow[l, "\mathsf{id}_A \otimes \alpha_{B,C,D}"']
\end{tikzcd}$$

$$\begin{tikzcd}
(A \otimes B) \otimes C \arrow[r, "\beta_{A,B} \otimes \mathsf{id}_C"] \arrow[d, "\alpha_{A,B,C}"] & (B \otimes A) \otimes C \arrow[d, "\alpha_{B,A,C}"] \\
A \otimes (B \otimes C) \arrow[d, "\beta_{A,B\otimes C}"] & B \otimes (A \otimes C) \arrow[d, "\mathsf{id}_B \otimes \beta_{A,C}"] \\
(B \otimes C) \otimes A \arrow[r, "\alpha_{B,C,A}"] & B \otimes (C \otimes A)
\end{tikzcd}$$

$$\begin{tikzcd}
(A \otimes I) \otimes B \arrow[rr, "\alpha_{A,I,B}"] \arrow[dr, "\rho_A"'] & & A \otimes (I \otimes B) \arrow[dl, "\lambda_B"] \\
& A \otimes B &
\end{tikzcd}
\qquad
\begin{tikzcd}
A \otimes B \arrow[d, "\beta_{A,B}"'] \arrow[dr, "\mathsf{id}_{A\otimes B}"] & \\
B \otimes A \arrow[r, "\beta_{B,A}"'] & A \otimes B
\end{tikzcd}$$

$$\begin{tikzcd}
I \otimes A \arrow[rr, "\beta_{I,A}"] \arrow[dr, "\lambda_A"'] & & A \otimes I \arrow[dl, "\rho_A"] \\
& A &
\end{tikzcd}$$

Monoidal categories posses additional structure, and hence, ordinary functors are not enough, thus, the notion must also be extended.

Definition 4. *Suppose we are given two monoidal categories* $(\mathcal{M}_1, \top_1, \otimes_1, \alpha_1, \lambda_1, \rho_1)$ *and* $(\mathcal{M}_2, \top_2, \otimes_2, \alpha_2, \lambda_2, \rho_2)$*. Then a* ***monoidal functor*** *is a functor* $F : \mathcal{M}_1 \longrightarrow \mathcal{M}_2$*, a map* $m_{\top_1} : \top_2 \longrightarrow F\top_1$ *and a natural transformation* $m_{A,B} : FA \otimes_2 FB \longrightarrow F(A \otimes_1$

B) subject to the following coherence conditions:

$$\begin{tikzcd}
(FA \otimes_2 FB) \otimes_2 FC \arrow[r, "\alpha_{2\,FA,FB,FC}"] \arrow[d, "m_{A,B} \otimes \mathsf{id}_{FC}"] & FA \otimes_2 (FB \otimes_2 FC) \arrow[d, "\mathsf{id}_{FA} \otimes m_{B,C}"] \\
F(A \otimes_1 B) \otimes_2 FC \arrow[d, "m_{A \otimes_1 B, C}"] & FA \otimes_2 F(B \otimes_1 C) \arrow[d, "m_{A, B \otimes_1 C}"] \\
F((A \otimes_1 B) \otimes_1 C) \arrow[r, "F\alpha_{1\,A,B,C}"] & F(A \otimes_1 (B \otimes_1 C))
\end{tikzcd}$$

$$\begin{tikzcd}
\top_2 \otimes_2 FA \arrow[r, "\lambda_{2\,FA}"] \arrow[d, "m_{\top_1} \otimes \mathsf{id}_{FA}"] & FA \\
F\top_1 \otimes_2 FA \arrow[r, "m_{\top_1, A}"] & F(\top_1 \otimes_1 A) \arrow[u, "F\lambda_{1\,A}"]
\end{tikzcd}
\qquad
\begin{tikzcd}
FA \otimes_2 \top_2 \arrow[r, "\rho_{2\,FA}"] \arrow[d, "\mathsf{id}_{FA} \otimes m_{\top_1}"] & FA \\
FA \otimes_2 F\top_1 \arrow[r, "m_{A, \top_1}"] & F(A \otimes_1 \top_1) \arrow[u, "F\rho_{1\,A}"]
\end{tikzcd}$$

B Source Sink Graphs are Symmetric Monoidal

In this appendix I show that the category of source-sink graphs defined by Jhawar et al. [4] is symmetric monoidal. First, recall the definition of source-sink graphs and their homomorphisms.

Definition 5. *A* ***source-sink graph*** *over* B *is a tuple* $G = (V, E, s, z)$*, where* V *is the set of vertices,* E *is a multiset of labeled edges with support* $E^* \subseteq V \times \mathsf{B} \times V$*,* $s \in V$ *is the unique start,* $z \in V$ *is the unique sink, and* $s \neq z$*.*

Suppose $G = (V, E, s, z)$ *and* $G' = (V', E', s', z')$*. Then a* ***morphism between source-sink graphs****,* $f : G \to G'$*, is a graph homomorphism such that* $f(s) = s'$ *and* $f(z) = z'$*.*

Suppose $G = (V, E, s, z)$ and $G' = (V', E', s', z')$ are two source-sink graphs. Then given the above definition it is possible to define sequential and non-communicating parallel composition of source-sink graphs where I denote disjoint union of sets by $+$ (p 7. [4]):

Sequential Composition :
$G \rhd G' = ((V \setminus \{z\}) + V', E^{[s'/z]} + E', s, z')$

Parallel Composition :
$G \odot G' = ((V \setminus \{s, z\}) + V', E^{[s'/s, z'/z]} + E', s', z')$

It is easy to see that we can define a category of source-sink graphs and their homomorphisms. Furthermore, it is a symmetric monoidal category were parallel composition is the symmetric tensor product. It is well-known that any category with co-products is symmetric monoidal where the co-product is the tensor product.

I show here that parallel composition defines a co-product. This requires the definition of the following morphisms:

$$\begin{aligned}
&\mathsf{inj}_1 : G_1 \to G_1 \odot G_2\\
&\mathsf{inj}_2 : G_2 \to G_1 \odot G_2\\
&\langle f, g\rangle : G_1 \odot G_2 \to G
\end{aligned}$$

In the above $f : G_1 \to G$ and $g : G_2 \to G$ are two source-sink graph homomorphisms. Furthermore, the following diagram must commute:

$$\begin{tikzcd}
& G \arrow[dl, "f"'] \arrow[dr, "g"] & \\
G_1 \arrow[r, "\mathsf{inj}_1"'] & G_1 \odot G_2 \arrow[u, "\langle f,g\rangle"] & G_2 \arrow[l, "\mathsf{inj}_2"]
\end{tikzcd}$$

Suppose $G_1 = (V_1, E_1, s_1, z_1)$, $G_2 = (V_2, E_2, s_2, z_2)$, and $G = (V, E, s, z)$ are source-sink graphs, and $f : G_1 \to G$ and $g : G_2 \to G$ are source-sink graph morphisms – note that $f(s_1) = g(s_2) = s$ and $f(z_1) = g(z_2) = z$ by definition. Then we define the required co-product morphisms as follows:

$$\begin{aligned}
&\mathsf{inj}_1 : V_1 \to (V_1 \setminus \{s_1, z_1\}) + V_2\\
&\mathsf{inj}_1(s_1) = s_2\\
&\mathsf{inj}_1(z_1) = z_2\\
&\mathsf{inj}_1(v) = v, \text{ otherwise}\\[1em]
&\mathsf{inj}_2 : V_2 \to (V_1 \setminus \{s_1, z_1\}) + V_2\\
&\mathsf{inj}_2(v) = v\\[1em]
&\langle f, g\rangle : (V_1 \setminus \{s_1, z_1\}) + V_2 \to V\\
&\langle f, g\rangle(v) = f(v), \text{ where } v \in V_1\\
&\langle f, g\rangle(v) = g(v), \text{ where } v \in V_2
\end{aligned}$$

It is easy to see that these define graph homomorphisms. All that is left to show is that the diagram from above commutes:

$$\begin{aligned}
(\mathsf{inj}_1; \langle f, g\rangle)(s_1) &= \langle f, g\rangle(\mathsf{inj}_1(s_1))\\
&= g(s_2)\\
&= s\\
&= f(s_1)
\end{aligned}$$

$$\begin{aligned}
(\mathsf{inj}_1; \langle f, g\rangle)(z_1) &= \langle f, g\rangle(\mathsf{inj}_1(z_1))\\
&= g(z_2)\\
&= z\\
&= f(z_1)
\end{aligned}$$

Now for any $v \in V_1$ we have the following:

$$\begin{aligned}(\mathsf{inj}_1; \langle f, g\rangle)(v) &= \langle f, g\rangle(\mathsf{inj}_1(v)) \\ &= f(v)\end{aligned}$$

The equation for inj_2 is trivial, because inj_2 is the identity.